REMARKS

1. The Double Patenting Rejection Should Be Withdrawn

The Examiner rejected claims 1-39 under the judicially created doctrine of obviousness type double patenting as unpatentable over claims 1-45 of U.S. Patent No. 6,631,516 ('516 patent). (Office Action, pg. 2) Applicants traverse for the following reasons.

Applicants submit that the non-statutory double patenting rejection should be withdrawn because the differences between the claims of the present application and the claims of the '516 patent are not a mere obvious variation. (Manual of Patent Examination and Procedure (MPEP), Sec. 804, pg. 800-22 (Rev. Feb. 2003)

The independent claims 1, 12, and 23 of the present application require that the language statements include declarations defining attribute information for a symbol or reference and that the attribute information included with the language statements are made available when binding object modules into the program object. The claims 1, 12, and 23 of the present application further require that the attribute information in the object module is derived from the language statements declaring the attribute information. Applicants submit that nowhere are these combination of requirements of the independent claims of the present application found in the claims of the 516 patent. Applicants submit that this distinction of the present claims over the claims of the '516 patent are not a mere obvious variation.

Accordingly, the non-statutory double patenting rejection should be withdrawn.

2. Claims 1, 3-7, 10-12, 14-18, 21-23, 25-29, and 35-39 are Patentable Over the Cited Art

The Examiner rejected claims 1, 3-7, 10-12, 14-18, 21-23, 25-29, and 35-39 as obvious (35 U.S.C. §103) over Lee (U.S. Patent No. 5,553,286) in view of Looney (U.S. Patent No. 6,366,876). Applicants traverse this rejection for the following reasons.

Claims 1, 12, and 23 concern producing an executable file for execution by a computer, comprising: receiving a plurality of programming language statements comprising a source program into the computer, wherein the language statements include declarations defining attribute information for symbol references and symbol definitions; translating the source program into an object module, wherein the object module is capable of including: a symbol reference; a symbol definition; attribute information for the symbol reference derived from the

language statements declaring the attribute information; and attribute information for the symbol definition derived from the language statements declaring the attribute information; binding object modules into a program object, wherein the attribute information is available when binding object modules into the program object; resolving in the program object an external symbol reference in the object module with an external symbol definition in another object module.

In the Office Action, the Examiner cited col. 15, line 66 to col. 16, line 36 of Looney as teaching the claim requirement that the the language statements include declarations defining attribute information for symbol references and symbol definitions. This attribute information defined by the language statements is available when binding object modules. (Office Action, pgs. 4 and 12-3) Applicants traverse this finding.

The cited cols. 15-16 of Looney mention that a class file contains information that represent references to other classes, fields, and methods that are stored in the classes "Constant Pool". Constants in this pool are descriptors that represent a reference to another class. The cited col. 16 further mentions that a class file refers to descriptors in tables. These tables include interfaces the class implements, fields in the class, and attribute tokens that describe access and descriptors. The class file also has symbolic information to identify dependency references.

Although the cited cols. 15-16 discuss descriptors and attributes maintained by class files, nowhere do the cited cols. 15-16 anywhere teach or suggest that the language statements, or class file statements, teach or suggest the claim requirements of a source program including declarations defining attribute information for a symbol reference or definition available when binding object modules in a program object. Although the cited cols. 15-16 of Looney discuses attribute information for methods in the class, as the Examiner notes, this cited attribute information is different from and does not teach or suggest the claimed attribute information of a symbol reference or definition available when binding object modules into the program object. Nowhere does the cited Looney anywhere teach, suggest or mention declarations defining attributes available when binding object modules as claimed.

Accordingly, these limitations of the independent claims distinguish over the cited art.

With respect to the other requirements of independent claims 1, 12, and 23, the Examiner repeated many of the grounds of rejection in the Final Office Action dated June 17, 2003.

Applicants submit that the Examiner did not respond or address the arguments Applicants presented to distinguish over these cited grounds. Applicants previously explained why the cited Lee does not teach the below discussed claim limitations. If the Examiner maintains this rejection of independent claims 1, 12, and 23, Applicants request that the Examiner respond specifically to the below arguments that explain clear deficiencies of the cited Lee.

The Examiner continues to cite to col. 6, line 51 to col. 7, line 1 and col. 8, lines 37-37 of Lee as teaching or suggesting the claim requirement that the object module include attribute information for the symbol reference derived from the language statements and attribute information for the symbol definition derived from the language statements. (Office Action, pg. 4) Applicants traverse for the following reasons.

The Examiner found that the index numbers, size and offset fields discussed in cited cols. 6-7 teach the claimed attribute information for the symbol reference and definition. The cited index, size and offset information of Lee concern the physical limitations of a load module. Lee discusses that source programs are compiled into object modules, which are linked together to form a load module, which is executable. (Lee, col. 6, lines 9-19). Thus, the cited size and offset fields concern physical limitations of the load module that are eliminated in the program object to allow accommodation of more sections and modules in the program object of Lee. Such cited information on the physical limitations of a program object with respect to the sections and modules included in the program object having nothing to do with and do not teach or suggest attribute information on symbol references and definitions included in the object module that is derived from the language statements that define such attribute information.

Further, this cited information of Lee on the physical layout of a program object does not teach or suggest the claimed attribute information for a symbol reference or definition that provides information defined in the source program language statements.

In the Office Action, the Examiner did not respond to Applicant's explanation that the claimed symbol reference or definition attribute information included in the object module is different from the cited offset and size fields that define physical limitations of how object modules may be included in the program object of Lee. If the Examiner continues the rejection, Applicants request that the Examiner explain how the cited offset and size fields of Lee that define physical limitations of how object modules may be included in the program object of Lee

teach or suggest the claimed attribute information for a symbol reference or definition that provides information defined in the source program language statements.

The Examiner also found that the pointers discussed in cols. 6-7 of Lee teaches the claimed symbol attribute information. (Office Action, pg. 4). Applicants traverse because a pointer is not attribute information on a symbol reference or definition defined in the language statements that declare such attribute information as claimed. Instead, a pointer is just a way to locate an item. Thus, the discussion in the cited col. 6 of replacing external names from a dictionary with pointers does not teach or suggest the claimed symbol attribute information that is derived from the language statements declaring and defining the attribute information.

Further, the index numbers the Examiner cites in col. 6, lines 55-65 also do not teach or suggest the claimed attribute information. (Office Action, pg. 4) These cited index numbers are an artifact of the program object and concern the size of numbers used to locate external names and sections. Such index numbers concerning the program object do not teach or suggest the claimed symbol attribute information that is derived from the language statements declaring the attribute information. Further, Applicants submit that the cited index numbers do not comprise the claimed attribute information that provides information defined in the source program language statements.

Moreover, the above discussed size, offset, pointers, and index numbers providing information on the layout of a program object cannot comprise the claimed attribute information because the claims require that the defined attribute information for the symbol reference or definition is included in the object module. Information on the layout of a program object in which object modules are included does not teach or suggest attribute information on a symbol included in an object module. If the Examiner maintains the rejection, Applicants request the Examiner to respond to this argument to explain how information on the layout of a program object does in fact teach or suggest attribute information for a symbol reference that is included in the object module.

The Examiner cited col. 7, lines 4-30 and col. 8, lines 37-46 as disclosing the claim requirement of binding object modules into a program object, wherein the attribute information is available when binding object modules into the program object. (Office Action, pg. 4) Applicants traverse.

The cited col. 8 discusses how the binder overlays items to form the single program object. This section discusses information the binder uses to construct the program object, including what is referred to as class attributes. The cited col. 7 discusses information used by the binder when constructing the program object, such as class attributes. Applicants submit that the cited information and techniques the binder uses to construct a program object nowhere teaches or suggests the claim requirement that attribute information on a symbol reference or definitions, derived from the language statements that declare and define such attribute information and included in the object module, is available to the binder when binding object modules into the program object. Instead, the cited attributes the binder uses to control the loading and binding of items in cols. 7 and 8 concern different types of information.

The Examiner did not address the above arguments in the Office Action. If the Examiner continues his rejection, Applicants request the Examiner to explain how information a binder uses to construct a program object teaches or suggest the claim requirement that attribute information on a symbol reference or definitions, derived from the language statements that declare and define such attribute information and included in the object module, is available to the binder when binding object modules into the program object.

Accordingly, claims 1, 12, and 23 are patentable over the cited art because the cited Lee does not teach or suggest all the claim requirements.

Claims 3-7, 10, 11, 13-18, 21, 22, 25-29, and 35-39 are patentable over the cited art because they depend from one of claims 1, 12, and 23, either directly or indirectly. The following dependent claims provide further grounds of patentability over the cited art

Claims 3, 14, and 25 depend from claims 1, 12, and 23 and further require that the object module is further capable of including fixed attribute information derived from language statements declaring attribute information for the symbol reference and symbol definition.

In the Office Action, the Examiner repeated citing col. 7, lines 4-30 and 47-49 and col. 9, lines 38-45 of Lee as teaching or suggesting the additional requirements of these claims and additionally cited col. 3, lines 14-15. (Office action, pg. 5) Applicants traverse.

The cited col. 7, lines 4-30 discuss class attribute information that controls the loading and binding of items in the class, including physical organization, how the items are to be structured during binding, whether the class can contain address constants, whether the class

should be loaded into storage, etc. The cited col. 7, lines 47-49 mentions that the binder logic is reduced by confining loading and binding variations to a finite set of class attributes.

As discussed, the cited col. 7 concerns information the binder uses when overlaying object modules into the program object of Lee. Nowhere does this cited col. 7 anywhere teach or suggest the claim requirement of the object module including fixed attribute information derived from language statements declaring attribute information for the symbol reference or definition that is available during binding.

The cited col. 9 mentions that the loader opens a file and that the object's structural data is read into memory. The loader allocates storage for each loadable segment in the object. The length, location and other characteristics of each block of storage will be determined by the class attributes. This cited section concerns how to load a program object and nowhere teaches or suggests the claim requirements that the attributes available to the binder include fixed attribute information derived from language statements declaring attribute information for a symbol reference or definition

The cited col. 3 mentions control records comprising a collection of the length and placement of the text record. Nowhere does the cited col. 3 anywhere teaches or suggests the claim requirements that the attributes available to the binder include fixed attribute information derived from language statements declaring attribute information for a symbol reference or definition.

In the Response to Arguments, the Examiner suggested that the new grounds of rejection mooted Applicant's explanations of the deficiencies of Lee. (Office Action, pg. 13) However, in rejecting claims 3, 14, and 25, the Examiner continued to cite Lee as teaching the additional requirements of these claims. Thus, Applicant's above explanation as to the deficiencies of the continued cited Lee are applicable and require Examiner response notwithstanding the new grounds of rejection.

Accordingly, claims 3, 14, and 25 provide additional grounds of patentability over the cited art because the cited Lee does not teach or suggest the additional dependent claim requirements.

Claims 5, 16, and 27 depend from claims 1, 12, and 23 and require that the object module is further capable of including an address constant for a symbol referenced in the module and

attribute information derived from language statements declaring attribute information for the address constant. The Examiner cited col. 8, lines 48-52 and col. 3, lines 16-18 of Lee as teaching the claim requirement of attribute information derived from language statements declaring attribute information for the address constant, where the attribute information is for symbol definitions and references. (Office Action, pg. 5). Applicants traverse.

The cited col. 3 discusses the length and placement of a text record. The cited col. 8 mentions that the binder stores the target address of a relocatable address constant and that class offsets are stored in the address constant. Although the cited Lee discusses address constants, nowhere does the cited col. 8 anywhere teach or suggest the claim requirement that attribute information for a symbol reference and definition is derived from language statements declaring attribute information for the address constant.

The Examiner further cited the relocation dictionary discussed in col. 3, lines 14-15 that includes the location and type of each address constant in a text record. (Office Action, pg. 5). This information maintained in a relocation directory on an address constant is different from and does not teach or suggest that the object module include attribute information for a symbol reference or definition derived from language statements declaring and defining such attribute information for the address constant.

Further, the cited relocation directory is a record in the load module. (Lee, col. 3, lines 2-20). Claims 5, 16, and 27 concern attribute information in the object module derived from the language statements, which is different from the records of the load module in which object modules are included.

In the Response to Arguments, the Examiner suggested that the new grounds of rejection mooted Applicant's explanations of the deficiencies of Lee. (Office Action, pg. 13) However, in rejecting claims 5, 16, and 27, the Examiner continued to cite Lee as teaching the additional requirements of these claims. Thus, Applicant's above explanation as to the deficiencies of the continued cited Lee are applicable and require Examiner response notwithstanding the new grounds of rejection.

Accordingly, claims 5, 16, and 27 provide additional grounds of patentability over the cited art because the cited Lee does not teach or suggest the additional dependent claim requirements.

Claims 6, 17, and 28 depend from claims 5, 16, and 27, respectively, and provide additional requirements concerning the requirement that address constants provide attribute information for the symbol references. Again, although the cited Lee discusses address constants, nowhere does Lee anywhere teach or suggest that the object module include attribute information derived from the language statements for an address constant.

Claims 7, 18, and 29 depend from claims 6, 17, and 28 and further require that resolving the symbol reference and definition further comprises performing a compatibility check using the attribute information, wherein a separate compatibility check is performed for each reference to a symbol for which there is a separate address constant and separate attribute information for each address constant.

The Examiner cited col. 8, lines 6-54 of Lee as teaching the claim requirement of resolving the symbol reference and definition further comprises performing a compatibility check using the attribute information. (Office Action, pg. 6) Applicants traverse.

The cited col. 8 discusses how a binder creates a program object by performing editing operation son included modules by renaming or deleting external symbols, deleting sections, etc. Following, if any external references are unresolved another phase is started. The cited col. 8 further discusses how the binder overlays modules and other items to form the single program object. This section discusses information the binder uses to construct the program object, including what is referred to as class attributes.

Nowhere does the cited col. 8 anywhere teach or suggest performing a compatibility check to resolve symbol references and definitions using the attribute information. Although the cited col. 8 discusses resolving references, nowhere is there any teaching or suggestion of performing a compatibility check to resolve symbol references using attribute information as claimed.

Accordingly, claims 7, 18, and 29 provide additional grounds of patentability over the cited art because the cited Lee does not teach or suggest the additional dependent claim requirements.

Claims 10, 21, and 32 depends from claims 1, 12, and 23 and further require resolving the symbol reference and definition further comprises performing a compatibility check using signature data for the symbol definition and symbol reference. The Examiner cited col. 12, lines

1-5 and col. 10, lines 3-42 of Looney as teaching the requirements of these claims. (Office Action, pg. 6)

The cited col. 12 mentions that a class is instantiated when a method is parsed and that the method includes the methods signature. The cited col. 10 mentions that a method includes a signature attribute identifying the method. The cited col. 10 further discusses an attribute that identifies a compliance status of a package, class or method. The compliance status may indicate that an entry has been modified. Although the cited col. 10 discusses a compliance status, nowhere does the cited col. 10 anywhere teach or suggest performing a compatibility check using signature data for a symbol definition and reference to resolve the symbol reference and definition in the object module.

The Examiner mentions that the compliance data of Looney can be used for compatibility checking during the binding/linking process. Applicants traverse because nowhere does the cited Looney teach, suggest or disclose performing such checking during binding to resolve symbol references and definitions as claimed.

Accordingly, claims 10, 21, and 32 provide additional grounds of patentability over the cited art because the cited Looney does not teach or suggest the additional dependent claim requirements.

Claims 11, 22, and 33 depend from 10, 21, and 32 and further require determining whether the compatibility check failed and processing the attribute information declared for the symbol reference and definition that failed the compatibility check to determine a cause of the incompatibility. The Examiner cited col. 10, lines 20-64, col. 2, lines 19-47, col. 2, line 59 to col. 3, line 6, and col. 12, lines 54-58 of Looney. (Office Action, pg. 7)

Applicants traverse as a review below of these cited sections reveals that no cited section teaches of suggests determining whether the compatibility check failed and processing the attribute information declared for the symbol reference and definition that failed the compatibility check to determine a cause of the incompatibility.

The cited cols. 2-3 discusses resolving references within the application. If a reference cannot be resolved, the merged specification is examined to determine whether the operating platform contains a programming resource that can be used to resolve the reference. If a programming resource is found, a compliance status is examined to determine whether the

reference is available for use. Applicants submit that nowhere does the cited cols. 2-3 teach or suggest processing attribute information made available during binding if there is a failure to resolve a reference to determine a cause of incompatibility. Instead, the cited cols. 2-3 looks at attribute information to find a programming resource that can be used to resolve the reference. Nowhere does the cited cols. 2-3 teach or suggest looking at attribute information to determine the cause of the compatibility. Instead, the cited cols. 2-3 uses the attribute information to find a programming resource that can be used for the resolution, not determining the cause of incompatibility as claimed.

The cited col. 10 discusses a compliance status of a package, class or method. A "required" status indicates that the associated entry is required to be in the application environment, the "optional" attribute indicates that the entry may or may not be included in the application environment, and "modified" status indicates that the entry has been changed. Nowhere does the cited col. 10 anywhere teach or suggest processing attribute information to determine the cause of the incompatibility if a compatibility check failed. There is no mention or suggestion anywhere in the cited col. 10 of these specific claim requirements.

Accordingly, claims 11, 22, and 33 provide additional grounds of patentability over the cited art because the cited Looney does not teach or suggest the additional dependent claim requirements.

Claims 34, 36, and 38 depend from claims 1, 12, and 23 and further require that the attribute information describes whether the symbol definition or symbol reference is to data or executable code. The Examiner cited col. 1, line 49 to col. 2, line 47 of Looney as teaching the additional requirements of these claims. (Office Action, pgs. 8-9) Applicants traverse.

The cited cols. 1-2 discusses analyzing applications for compatibility with the intended device and determine whether a software application is compatible with an operating platform. This cited section discusses a grammar to specify programming resources of the operating platform that may be used to resolve an application's references.

Although the cited cols. 1-2 discuss providing information to use to resolve references, nowhere does this cited cols. 1-2 anywhere teach or suggest that the attribute information describes whether the symbol definition or symbol reference is to data or executable code. This cited section discusses certain compliance statuses, but none of these teach or suggest the specific claim

requirements. If the Examiner maintains this rejection, Applicants request the Examiner point to the specific section of Looney that teaches or suggest this specific requirement of the attribute information.

Claims 35, 37, and 39 depend from claims 1, 12, and 23 and further require that the attribute information indicates data types and structures if the symbol definition or symbol reference is for data and indicates a number and types of arguments passed or received if the symbol definition or symbol reference is for executable code.

The Examiner asserts that Lee and Looney teach or suggest the specific requirements of these claims. (Office Action, pg. 9) However, the Examiner has not cited any specific section of Lee and Looney that teaches, suggests or mentions the requirement that the attribute information indicates data types and structures if the symbol definition or symbol reference is for data and indicates a number and types of arguments passed or received if the symbol definition or symbol reference is for executable code, where the attribute information is declared with the source program and made available during binding. If the Examiner maintains this rejection, Applicants request the Examiner point to the specific section of Looney that teaches or suggest this specific requirement of the attribute information.

3. Claims 2, 4, 8, 9, 13, 15, 19, 20, 24, 26, 30, and 31 are Patentable Over the Cited Art

The Examiner rejected claims 2, 8, 9, 13, 19, 20, 24, 30, and 31 as obvious (35 U.S.C. §103) over Lee and Looeny in view of Fitzgerald (U.S. Patent No. 5,408,665). Applicants traverse for the following reasons.

Claims 2, 4, 8, 9, 13, 15, 19, 20, 24, 26, 30, and 31 are patentable over the cited combination because they depend either directly or indirectly from one of claims 1, 12, and 23, which are patentable over the cited art for the reasons discussed above.

In rejecting these claims, the Examiner continues to rely on Fitzgerald as teaching or suggesting the additional dependent claim requirements. However, the Examiner stated that attacking Fitzgerald or Lee separately would be inapposite in view of the new combination with Looney (Office Action, pg. 13) Applicants traverse this finding because Applicants are pointing out why the cited Fitzgerald does not teach the additional dependent claim 2, 4, 8, 9, 13, 15, 19, 20, 24, 26, 30, and 31 requirements. Further, where the Examiner relies on Looney for these additional

claim requirements, Applicants explain why the cited Looney is also deficient and does not teach or suggest these dependent claim requirements to overcome the deficiencies of Fitzgerald addressed below.

Claims 2, 13, and 24 depend from claims 1, 12, and 23 and further require that the language statement is capable of indirectly declaring extended attribute information defined in another location in the object module.

The Examiner cited the package 816 and MethodRef class 412of Looney as teaching class attribute declaration as claimed. (Office Action, pg. 10) The cited MethodRef class 412 is part of specification references that form a representation of a file and includes a reference to a parent. (Looney, Col. 11) The cited package 816 has a reference for each class file. (Looney, Col. 17) Nowhere do these cited elements of Looney anywhere teach or suggest that a language statement is capable of indirectly declaring extended attribute information defined in another location in the object module.

The Examiner further stated that declaring a class is equivalent to declaring a reference object. (Office Action, pg. 10) Applicants traverse and submit that mentioning declaring a class as Looney does nowhere teaches or suggests the specific claim requirement of indirectly declaring extended attribute information defined in another location in the object module. The Examiner is modifying the cited Looney in a manner nowhere taught or suggested in the cited art or explained.

The Examiner further repeated his findings from the Final Office Action in finding that Fitzgerald teaches the additional requirements of these claims. (Office action, pg. 10) Applicants traverse.

The cited cols. 10-11 of Fitzgerald discuss an extended dictionary that includes information on object modules in a library to locate the object modules in the library. Applicants submit that this cited information in Fitzgerald to locate object modules using an extended dictionary, such as the Module ID, does not teach or suggest the claim requirement that the language statement from which the symbol attribute information is derived is indirectly declared in another location in the object module. In other words, information in the extended dictionary used to locate object modules in a library is different and does not teach or suggest the claimed extended attribute information declared in the language statements and defined in another location of the object module.

If the Examiner maintains this rejection in view of Fitzgerald for the additional requirements of claim 2, Applicants request that the Examiner specifically show where Fitzgerald or Looney, alone or in combination, teaches or suggests that language statements in the source program is capable of indirectly declaring extended attribute information defined in another location in the object module

Accordingly, claims 2, 13, and 24 provide additional grounds of patentability over the cited art.

Claims 8, 19, and 30 depend from claims 1, 12, and 23 and further require the object module further includes an External Symbol Directory (ESD) including a record capable of indicating a symbol in the program, a location of the symbol in the program, and a pointer to attribute information in the program for the symbol.

The Examiner cited the package 816 and MethodRef class 412 of Looney as teaching class attribute declaration as claimed. (Office Action, pg. 11) The cited MethodRef class 412 is part of specification references that form a representation of a file and includes a reference to a parent. (Looney, Col. 11) The cited package 816 has a reference for each class file. (Looney, Col. 17) Nowhere do the cited elements of Looney anywhere teach or suggest that the object module further includes an External Symbol Directory (ESD) including a record capable of indicating a symbol in the program, a location of the symbol in the program, and a pointer to attribute information in the program for the symbol.

The Examiner is modifying the cited Looney in a manner nowhere taught or suggested in the cited art or explained.

The Examiner repeated his findings from the Final Office Action that the External Dictionary of Fitzgerald teaches the claim requirement that the ESD include a pointer to attribute information in the program for the symbol. (Office Action, pg. 11) Applicants traverse.

As discussed, the cited External Dictionary of Fitzgerald includes information to locate modules. The Examiner has not cited any part of Fitzgerald that teaches or suggests the claim requirement of a pointer to attribute information for a symbol definition or reference, where such attribute information is derived from language statements that include declarations of attribute information that provides information defined in the source program language statements. Instead, the cited External Dictionary concerns information on how to locate modules. Fitzgerald discusses

how this External Dictionary is used by a linker to locate object modules to link together. (Fitzgerald, col. 8, line 62 to col. 10, line 9). Fitzgerald discusses how the Extended Dictionary allows object modules to be located during the passes when linking. (Fitzgerald, col. 9, line 49 to col. 10, line 9). The cited figures 4b, 5b, 6b, c, d concern steps to process the extended dictionary during linking passes to process the object modules. Nowhere does the cited Fitzgerald anywhere teach or suggest the claimed pointer to attribute information for a symbol definition or reference as claimed.

If the Examiner maintains this rejection in view of Fitzgerald and Looney for the additional requirements of claim 2, Applicants request that the Examiner specifically show where Fitzgerald or Looney, alone or in combination, teaches or suggests a pointer to attribute information for a symbol definition or reference, where such attribute information is derived from language statements that include declarations of attribute information that provides information defined in the source program language statements.

Accordingly, claims 8, 19, and 30 provide additional grounds of patentability over the cited art.

Claims 9, 20, and 31 depend from claims 1, 12, and 23 and further require that the object module further includes a Relocation List Directory (RLD) including a record capable of indicating the location of the referenced symbol, a location of an address constant for the referenced symbol in the program, and a pointer to attribute information for the address constant in the program.

The Examiner again cited the package 816 and MethodRef class 412 of Looney as teaching class attribute declaration as claimed. (Office Action, pg. 11) The cited MethodRef class 412 is part of specification references that form a representation of a file and includes a reference to a parent. (Looney, Col. 11) The cited package 816 has a reference for each class file. (Looney, Col. 17) Nowhere do the cited elements of Looney anywhere teach or suggest that the object module further includes a Relocation List Directory (RLD) including a record capable of indicating the location of the referenced symbol, a location of an address constant for the referenced symbol in the program, and a pointer to attribute information for the address constant in the program.

The Examiner is modifying the cited Looney in a manner nowhere taught or suggested in the cited art or explained.

The Examiner again cited the External Dictionary of Fitzgerald discussed above as teaching the claim requirement a pointer to attribute information for the address constant in the program. (Final Office Action, pgs. 6-7) Applicants traverse.

The Examiner cited step 603 in FIGs. 6A. This step is part of the processing of the Extended Dictionary to locate an object module and to obtain an ID of the module needed by the current module. (Fitzgerald, col. 15, lines 35-40, as well as other steps mentioned in cols. 14-15)

Nowhere does the cited Fitzgerald anywhere teach or suggest the claim requirement of a pointer in the Relocation List Directory (RLD) to attribute information for the address constant in the program, where such attribute information is derived from language statements that include declarations of attribute information that provides information defined in the source program language statements.

Accordingly, claims 9, 20, and 31 provide additional grounds of patentability over the cited art.

4. Conclusion

For all the above reasons, Applicant submits that the pending claims 1-39 are patentable over the art of record. Applicants submit that no additional fees are needed. Nonetheless, should any additional fees be required, please charge Deposit Account No. 09-0460.

The attorney of record invites the Examiner to contact him at (310) 553-7977 if the Examiner believes such contact would advance the prosecution of the case.

Dated: March 2, 2004

By: _____
David W. Victor
Registration No. 39,867

Please direct all correspondences to:

David Victor
Konrad Raynes, & Victor, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: 310-553-7977
Fax: 310-556-7984